Module 5
Strings and Text Files

Tamer Abdou, PhD

IND 830 - Python Programming for Data Science

Data Science Laboratory

Ryerson University

Ryerson University
The Chang School of Continuing Education

Outline

1 Characters and Strings

2 Text Files

Ryerson University
The Chang School of Continuing Education

Characters and Strings

1 Characters and Strings
Accessing Characters and Substrings
Strings and Number Systems
String Methods

2 Text Files

The Structure of Strings

- A **string** is a data structure that consists of several smaller pieces of data
- A string is a sequence of zero or more characters
- The positions of those characters are numbered from 0 to the length of the string minus one
- The `len()` function can be used to return the number of characters a string contains

The Subscript Operator

```
<a string> [<an integer expression>]
```

- The first part is the inspected string
- The second part in brackets indicates the position of a particular character in the inspected string
- The integer expression is also called an **index**

Example 5.1 (Script)

```
1  courseCode = "CIND 830"
2  #Display the length of the string which is '8'
3  print(len(courseCode))
4  #Display the 1st character which is 'C'
5  print(courseCode[0])
6  #Display the 7th character which is '3'
7  print(courseCode[6])
8  #Display the last character which is '0'
9  print(courseCode[len(courseCode)-1])
10 print(courseCode[-1])
```

Slicing Strings

- A **subscript** is a portion of a string that can be extracted
- Python's subscript operator can be used to obtain a substring through a process called 'slicing'
- To extract a substring, a colon '**:**' is used and an integer value can appear on either side of the colon
- If two integer positions are included in the slice (e.g. $[x : y]$)
  - The range extends from x up to $y - 1$
  - If x **is omitted**, all the characters extending to the **beginning** will be included in the substring
  - If y **is omitted**, all the characters extending to the **end** will be included in the substring

Example 5.2 (Script)

```
1  progLang = "Python"
2  #Display the first two characters which are "Py"
3  print(progLang[ 0 : 2 ])
4  print(progLang[   : 2 ])
5  print(progLang[   :-4 ])
6  #Display the last four characters which are "thon"
7  print(progLang[ 2 : 6 ])
8  print(progLang[ 2 :   ])
9  print(progLang[-4 :   ])
10 #Display all the characters
11 print(progLang[ 0 : 6 ])
12 print(progLang[   : 6 ])
13 print(progLang[ 0 :   ])
14 print(progLang[   :   ])
```

Strings and Number Systems

- Any binary number can be represented by a string of bits
- To determine the integer quantity that a string of bits represents, we multiply the value of each bit (0 or 1) by its positional value, then add the results
- $101_2 = 1 * 2^2 + 0 * 2^1 + 1 * 2^0$
- $1011_2 = 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0$

Example 5.3 (Script)

```
1  #Convert 1011 to integer
2  bitString = "1011"
3  result = 0
4  exponent = len(bitString) - 1
5  for bit in bitString:
6      result = result + int(bit) * 2 ** exponent
7      exponent = exponent - 1
8  print("The integer value is", result)
```

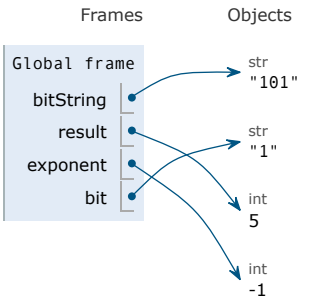
Example 5.3 (Program Execution)

Write code in Python 3.6 (drag lower right corner to resize code editor)

```
1  #Convert 101 to integer
2  bitString = "101"
3  result = 0
4  exponent = len(bitString) - 1
5  for bit in bitString:
6      result = result + int(bit) * 2 ** exponent
7      exponent = exponent - 1
8  print("The integer value is", result)
```

Print output (drag lower right corner to resize)

```
('The integer value is', 5)
```



line that has just executed
next line to execute

String Methods

```
<an object>.<method name>(<argument 1>, ..., <argument n>)
```

- A method can be invoked by an **object**
- The object needs to be placed before the name of the method
- Methods can have arguments and return values. For example:
  - The `split()` method creates a list of words based on a specified separator
  - The `isalpha()` returns `True` if the sentence contains only letters or `False` otherwise
  - The `upper()` method returns an uppercase version of a given sentence or string
  - The `count()` method returns the total number of non-overlapping occurrences of a given substring

Example 5.4 (Script)

```
1  #Practicing String Methods
2  Sentence = '''String literals are written in a variety of ways:
3  Single, Double, and Triple quotes'''
4
5  print(Sentence.split())         #Split without a delimiter
6  print(len(Sentence.split()))
7
8  print(Sentence.split("\n"))     #Split using a delimiter
9  print(len(Sentence.split(":")))
10
11 print(Sentence.upper())         #Uppercase or lowercase
12 print(Sentence.lower())
13
14 print(Sentence.isalpha())       #Letters or Digits?
15 print(Sentence.isdigit())
16
17 print(Sentence.count("le"))     #Number of occurrences of "le"
```

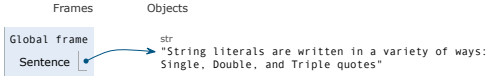
Example 5.4 (Program Execution)

Write code in Python 3.6 (drag lower right corner to resize code editor)

```
1  #Practicing String Methods
2  Sentence = '''String literals are written in a variety of ways:
3  Single, Double, and Triple quotes'''
4
5  print(Sentence.split())        #Split without a delimiter
6  print(len(Sentence.split()))
7
8  print(Sentence.split("\n"))    #Split using a delimiter
9  print(len(Sentence.split(":")))
10
11 print(Sentence.upper())        #Uppercase or lowercase
12 print(Sentence.lower())
13
14 print(Sentence.isalpha())      #Letters or Digits?
15 print(Sentence.isdigit())
16
17 print(Sentence.count("le"))    #Number of occurrences of "le"
```

Print output (drag lower right corner to resize)

```
['String', 'literals', 'are', 'written', 'in', 'a', 'variety', 'of', 'ways:', 'Single,', 'Double,', 'and', 'Triple', 'quotes']
14
['String literals are written in a variety of ways:', 'Single, Double, and Triple quotes']
2
STRING LITERALS ARE WRITTEN IN A VARIETY OF WAYS:
SINGLE, DOUBLE, AND TRIPLE QUOTES
string literals are written in a variety of ways:
single, double, and triple quotes
False
False
3
```



Ryerson University
The Chang School of Continuing Education

Text Files

1 Characters and Strings

2 Text Files
Reading and Writing Files
Accessing and Manipulating Files

Reading/Writing Text from/to a File

- Data can be input or output to a text file using a file object
- The `open()` function opens a connection to a specific file with an adjusted `mode` parameter
- The mode parameter '**r**' is for input/reading files and '**w**' for output/writing files
- String data can be read using the `read()` method, and written using the `write()` method
- Any opened file should be closed after finishing writing the data in it, and that can be achieved using the `close()` method
- Failure to close an opened file may result in data loss

Ryerson University The Chang School of Continuing Education

Example 5.5 (Script)

```
1  #Text file Operations and Methods
2
3  textFile = open("myfile.txt", "w")  #Opening a file for writing
4  textFile.write("First line.\nSecond line.\n")
5  textFile.close()
6
7  textFile = open("myfile.txt", "a")  #Opening a file for appending
8  text = ["Third line.\n"]
9  textFile.writelines(text)
10 textFile.close()
11
12 textFile = open("myfile.txt", "r")  #Opening a file for reading
13 contentsFile = textFile.read()      #Read and Store
14 print(contentsFile)                 #Display the contents
15 textFile.close()
```

Reading/Writing Numbers from/to a File

- All file input and output operations return data as strings
- Functions `int()` and `float()` are used to convert data items to integers and floating-point numbers respectively
- A simple `for-loop` can be used to access one line of text at a time
- A programmer can use the `split()` method to obtain a list of substrings then convert them into either integers or floating-point numbers

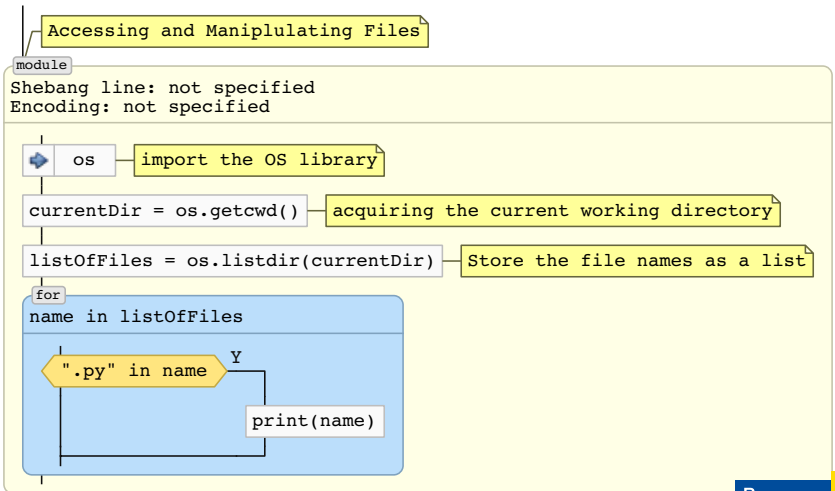
Accessing Files on Disks

- When interacting with files and before attempting to open them, it is preferable to check the existence of those files with the given path names
- The file system function `os.path.exists()` can be used to check the existence of a file
- To return or change the path of the current working directory, the `getcwd()` or `chdir(path)` function can be used
- The `mkdir()` function is used to create folders, and the `rmdir()` function is used to remove folders

Example 5.6 (Script)

```
1  #Accessing and Maniplulating Files
2
3  import os                        #import the OS library
4
5  currentDir = os.getcwd()     #acquiring the current working directory
6
7  listOfFiles = os.listdir(currentDir) #Store the file names as a list
8
9  for name in listOfFiles:
10     if ".py" in name:
11         print(name)
```

Example 5.6 (Control Flow)



References

- Lambert, K. A. (2010). Strings and text files. In *Fundamentals of Python: From first programs through data structures* (pp. 121-157). Boston, MA: Cengage Learning

- Python Software Foundation. (2020). Python 3.7.8 documentation. Retrieved from `https://docs.python.org/3.7/index.html`